Name: Arvind Ratnu

Email address: arvind.iiita07@gmail.com

Contact number: 6375159703

Anydesk address: 286373175

Years of Work Experience: 2.5 years

Date: 22th Dec 2021

Self Case Study -2: CommonLit Readability

"After you have completed the document, please submit it in the classroom in the pdf format."

Please check this video before you get started:
https://www.youtube.com/watch?time_continue=1&v=LBGU1_JO3kg

Overview

*** Write an overview of the case study that you are working on. ***(MINIMUM 200 words)*** ***

1. This case study is on the kaggle competition "**CommonLit Readability**" .Rate the complexity of literary passages for grades 3-12 classroom use.
2. In this competition, we're predicting the reading ease of excerpts from literature. We've provided excerpts from several time periods and a wide range of reading ease scores. Test set includes a slightly larger proportion of modern texts (the type of texts we want to generalize to) than the training set.
3. **The Dataset** consists of a training set(train.csv), test set(test set) and a sample submission file in the correct format(sample_submission.csv).
4. **The Dataset** has following columns: id(unique ID for excerpt), url_legal(URL of source - this is blank in the test set), license(license of source material - this is blank in the test set), excerpt(text to predict reading ease of), target(reading ease), standard_error(measure of spread of scores among multiple raters for each excerpt. Not included for test data) .
5. Submissions are scored on the root mean squared error.

6. The **target value** is the result of a Bradley-Terry analysis of more than 111,000 pairwise comparisons between excerpts. Teachers spanning grades 3-12 (a majority teaching between grades 6-10) served as the raters for these comparisons.
7. Standard error is included as an output of the Bradley-Terry analysis because individual raters saw only a fraction of the excerpts, while every excerpt was seen by numerous raters. The test and train sets were split after the target scores and standard error were computed.
8. This is a **NLP problem** where we have to try a large number of NLP models like BERT, RoBERTa, DeBERTa etc. .Then blend them together to arrive at the solution.

Research-Papers/Solutions/Architectures/Kernels

*** Mention the urls of existing research-papers/solutions/kernels on your problem statement and in your own words write a detailed summary for each one of them. If needed you can include images or explain with your own diagrams. ***

1. https://arxiv.org/abs/1810.04805

   Research Paper on BERT. Paper explains BERT and how pre-trained BERT models can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks without substantial task-specific architecture modifications.

2. https://www.kaggle.com/ratan123/in-depth-guide-to-google-s-bert

   Kaggle notebook explaining BERT with code.It covers all the topics which helps to understand BERT clearly.And, how to implement BERT for disaster NLP.

3. https://www.kaggle.com/yassinealouini/roberta-meets-tpus

   Kaggle notebook explaining roberta with code. It describes and explains how the model works. Then It builds and trains a model using the Tweet Sentiment Extraction dataset.

4. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15775971.pdf

   Research Paper on applying Ensembling Methods to BERT to Boost Model Performance. In this paper, they present various ensemble models that leverage BiDAF and multiple different BERT models to achieve superior performance. They also studied the effects on how dataset manipulations and different ensembling methods can affect the performance of models.

5. https://arxiv.org/abs/1905.05583

   In this research paper, they conduct exhaustive experiments to investigate different fine-tuning methods of BERT on text classification tasks and provide a general solution for BERT fine-tuning. The proposed solution obtains new state-of-the-art results on eight widely-studied text classification datasets.

6. https://arxiv.org/abs/2006.03654

   Research paper on DeBERTa (Decoding-enhanced BERT with disentangled attention) that improves the BERT and RoBERTa models.

7. https://machinelearningmastery.com/how-to-use-nelder-mead-optimization-in-python/

   Tutorial on the Nelder-Mead optimization algorithm which is a widely used approach for non-differentiable objective functions.The tutorial explains how to apply the Nelder-Mead algorithm for function optimization in Python and interpret the results of the Nelder-Mead algorithm on noisy and multimodal objective functions.

8. https://www.topbots.com/leading-nlp-language-models-2020/

   Blog has summarized various research papers featuring the key language models introduced during the last few years. The content includes original abstract, summary, core idea of paper, key achievement, AI community viewpoint, future research areas, possible business applications and implementation code.

9. https://insights.daffodilsw.com/blog/top-5-nlp-language-models

   This blog discusses top 10 pre-trained NLP models.

First Cut Approach

*** Explain in steps about how you want to approach this problem and the initial experiments that you want to do. ***(MINIMUM 200 words)*** ***

1. In this problem we need to try different and large numbers of NLP models like BERT, RoBERTa ,DeBERTa, DistilBERT, ELECTRA, GPT2, ALBERT etc.
2. We also will be trying different fine tuning strategies like Layer reinitialization,layer freeze,LR scheduler; and different learning rate,different attention heads,different dropout values,different epochs,different batch sizes etc.
3. Then we need to come with ensemble architecture. Here we need to try different ways like simple average, weighted average,log averages,median and power averaging.
4. We also will be trying the nelder-mead algorithm to find different weights to combine various nlp models. We will use both positive weight and negative weight given by the nelder-mead algorithm with restriction of weight summing to 1 being removed. Then we will adjust weights based on cv score and leaderboard score to find best weights.In the post process, we will multiply different coefficients depending on the predicted value. The threshold value will be determined by looking at the distribution of the target of the train.
5. At this point we should get good results. But if we are still not getting desirable results then we need to use external data and try pseudo labelling .The reason for this is the training data is small and we will use external data to increase training data. In this approach,we need to find a corpus of text (simplewiki, wikipedia, bookcorpus etc.) that looks somewhat like the target data set.Then we will make text snippets from external data that are approximately the same size as data in the train set.After that for each excerpt in train set we will use some paraphrase model to generate sentence embeddings and find text snippets with highest cosine similarity. Then we use the concept of pseudo labelling. In this we will use confident labelled external data to increase training data .Then use new training data to make predictions.

Notes when you build your final notebook:

1. You should not train any model either it can be a ML model or DL model or Countvectorizer or even simple StandardScalar
2. You should not read train data files
3. The function1 takes only one argument “X” (a single data points i.e 1*d feature) and the inside the function you will preprocess data point similar to the process you did while you featurize your train data
    a. Ex: consider you are doing taxi demand prediction case study (problem definition: given a time and location predict the number of pickups that can happen)
    b. so in your final notebook, you need to pass only those two values
    c. def final(X):

preprocess data i.e data cleaning, filling missing values etc

compute features based on this X

use pre trained model

return predicted outputs

final([time, location])

    d. in the instructions, we have mentioned two functions one with original values and one without it
    e. final([time, location]) # in this function you need to return the predictions, no need to compute the metric
    f. final(set of [time, location] values, corresponding Y values) # when you pass the Y values, we can compute the error metric(Y, y_predict)

4. After you have preprocessed the data point you will featurize it, with the help of trained vectorizers or methods you have followed for your train data
5. Assume this function is like you are productionizing the best model you have built, you need to measure the time for predicting and report the time. Make sure you keep the time as low as possible
6. Check this live session: https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4148/

hands-on-live-session-deploy-an-ml-model-using-apis-on-aws/5/module-5-feature-engineering-productionization-and-deployment-of-ml-models